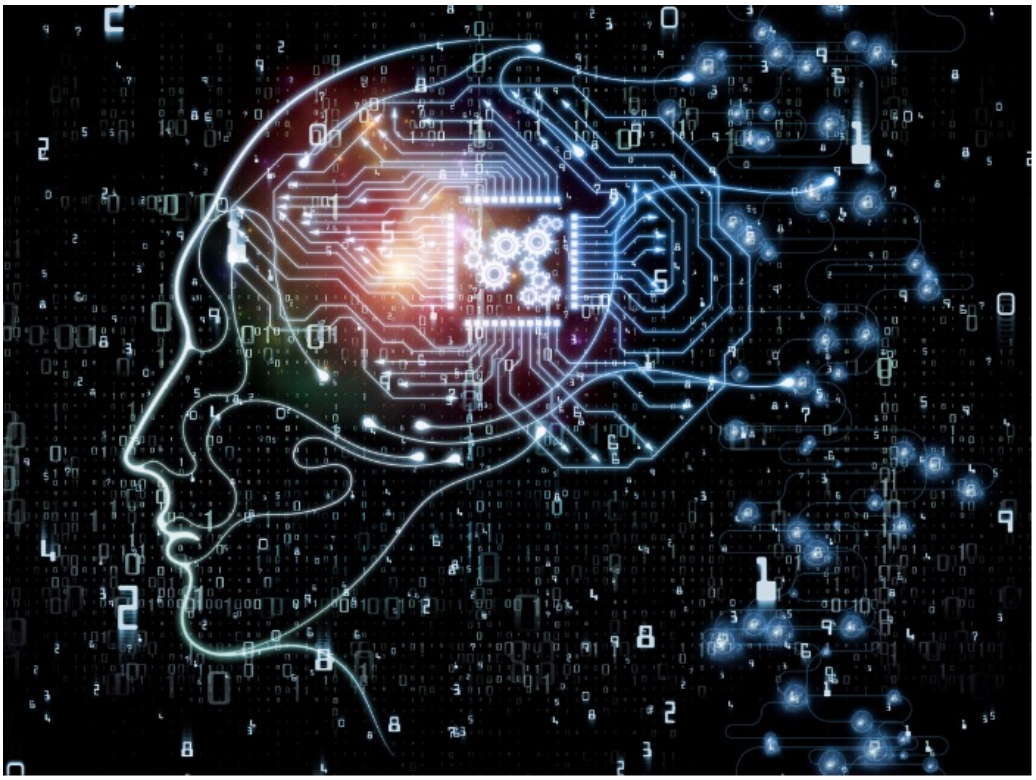
Universidad de Alicante

Robots Móviles

Trabajo final: Inteligencia Artificial aplicada a la robótica móvil



Autores:
Alejandro Correa Svensson y Mónica Lorenzo García.

20 de Diciembre de 2022

Índice

1. Inteligencia artificial en la robótica

La IA es un campo que surgió en la conferencia de Dartmouth en 1956. La inteligencia artificial supone una capacidad para comprender el entorno, ser capaz de encontrar el significado de las cosas y darles un sentido y para poder tomar decisiones en base a su conocimiento o mejorar en esta toma de decisiones en base a su experiencia.

Dentro de la Inteligencia Artificial se distinguen 2 tipos:

1. **Inteligencia Artificial Fuerte:** se le proporciona a un computador suficiente capacidad de procesamiento y se le da la suficiente inteligencia, se puede llegar a crear un ordenador que sea capaz de pensar y ser consciente de la misma manera que lo son los humanos. Este tipo de inteligencia aún no se ha conseguido ni espera poder alcanzarla en las próximas décadas.

2. **Inteligencia Artificial Débil:** El computador utiliza el comportamiento inteligente para resolver problemas complejos. Es importante saber que un comportamiento inteligente, por parte de una máquina, no implica que esta sea inteligente, es decir, no garantiza que tenga conciencia.

2. Aprendizaje por Refuerzo

El aprendizaje por refuerzo o Reinforcement Learning (RL) es una rama del Machine Learning donde la finalidad es conseguir que la máquina aprenda basándose en el esquema que se observa el la Figura 1. Este tipo de aprendizaje surge a partir de la necesidad de crear un algoritmo que fuera capaz de llevar a cabo acciones en un entorno que está afectado por múltiples variables que van cambiando con el tiempo.

Por un lado tenemos el agente, el cual se encarga llevar a cabo la acción gracias a una función de toma de decisiones también llamada política. Esta política de aprendizaje se va construyendo a la vez que el agente va adquiriendo conocimiento. Seguidamente, el entorno es el espacio en el que se encuentra el agente. El entorno se encuentra en cada instante de tiempo bajo un estado. También se encarga de proporcionar una recompensa o penalización al agente por la acción realizada. Por lo que de esta manera se consigue el *feedback* necesario para que el agente pueda aprender y mejorar en su toma de decisiones futuras.

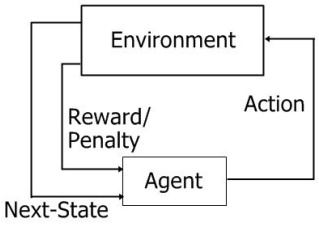


Figura 1: Esquema de aprendizaje por refuerzo

Para poder cuantificar la recompensa se aplica una función que lo cuantifique, siendo sus parámetros de entrada el estado en el que se encuentra el agente y la acción.

2.1. Aprendizaje por refuerzo en la robótica móvil

2.1.1. Procesos de Decisión de Markov (MDP):

Antes de explicar el Proceso de Decisión de Markov se debe hablar previamente de las cadenas de Markov. Las Cadenas de Markov son un proceso estocástico [1] en el que si el estado actual y los estados previos son conocidos, la probabilidad del estado futuro no depende de los estados anteriores, si no que solo depende del estado actual.

$$P(X_{n+1} = s_{n+1} \mid X_1 = s_1, X_2 = s_2, \ldots, X_n = s_n) =$$
$$= P(X_{n+1} = s_{n+1} \mid X_n = s_n)$$

Figura 2: Formulación de las cadenas de Markov.

Por consiguiente, el Proceso de Decisión de Markov es una extensión de las cadenas de Markov que modela un problema de decisión donde la dinámica del sistema viene dada por una función de probabilidad la cual mapea estados y acciones a otros estados. Es decir, es una sucesión de estados que cumplen la Cadena de Markov.
Matemáticamente un MDP es una tupla formada por un conjunto de estados (S), un conjunto finito de acciones que dependen de cada estado (A), una función de recompensa (R) y una función de transición de estados que viene dada como una distribución de probabilidad (Φ). Como se ha comentado anteriormente, la toma de decisiones viene dada por la política (π) que sera una tabla que indicara al agente *como actuar*. A partir de dicha política se define la función de valor de estados y la función de valor de acciones, las cuales dan forma a la ecuaciones de Bellman. [2]

$$M =< S, A, \Phi, R >$$

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s], \text{correspondiente a la función estado-valor}$$

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a], \text{correspondiente a la función acción-valor}$$

Por lo tanto la solución del problema del Aprendizaje por Refuerzo aplicando MDP se basa en la optimización de estas funciones. De esta manera, un buen agente inteligente es aquel que es capaz de predecir cual sera la consecuencia de cada opción, antes de ser tomada, consiguiendo así tomar las acciones que mas optimicen la solución.

2.1.2. Q-Learning

El Q-Learning se utiliza para optimizar la toma de decisiones de un agente, como ya se ha comentado antes, el objetivo final seria poder conocer la recompensa que se puede obtener realizando cada acción posible, sin realizarla de verdad.

Dado que esta es la idea básica, pero no es posible conocer con certeza absoluta el resultado de una acción, en Q-Learnig se utiliza la función de valor (Q) que nos devuelve la recompensa esperada al realizar una acción. Por medio de esto, conseguimos que la política a seguir sea la de escoger la acción que más valor devuelva, en función de (Q).

[1] Proceso Estocástico: concepto matemático que sirve para representar magnitudes aleatorias que varían con el tiempo.
[2] Ecuaciones de Bellman o también conocidas como ecuaciones de programación dinámica son fundamentales para resolver un problema de Aprendizaje por Refuerzo, ya que muchos de los algoritmos empleados para entrenar al agente tienen como objetivo calcular el valor de un estado o de una acción partiendo de estas ecuaciones.

Esta función (Q) se ira actualizando de la siguiente manera.

$$Q(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot \max_a Q(s_{t+1}, a))$$

Sabiendo esto, la tabla de la política del agente, se ira actualizando según el mejor valor que encuentre el proceso de Q-Learnig para cada par(estado(S),acción(A)). Cuando se esta realizando este proceso de aprendizaje, necesitamos una estrategia que nos ayude a elegir entre, explotar una opción, es decir, seleccionar la opción que mas valor aporte, o explorar, seleccionar una opción al azar.

Esta cuestión es muy importante ya que un correcto balance entre las dos, hará que visitemos el mayor numero de opciones, aunque muchas de estas no sean las mas óptimas a priori(exploración), mientras que no desistimos de la búsqueda de la función mas óptima(explotación).

La manera mas sencilla de hacer esto es con un comportamiento al azar, pero para obtener un mejor resultado se suele utilizar Epsilon-Greedy (ϵ). Para cada instante, se tomara la opción aleatorio con probabilidad (ϵ), y la opción con mayor posibilidad en caso contrario.
Definiendo el valor de (ϵ) para que aumente la probabilidad de exploración al comienzo del algoritmo, y vaya balanceando esa probabilidad hacia la explotación según avanza este.

A continuación se muestra un simple ejemplo de esta metodología.

```
1  import random
2  import numpy as np
3
4  def qlearn():
5      gamma = 0.99
6      alpha = 0.1
7      q = np.ones((5, 6))
8      q[4,:] = 0
9  # terminal state
10
11     for k in range(1, 1000):
12         s = 0
13         epsilon = 1/k
14         while True:
15             if random.random() < epsilon:
16                 a = random.randint(0, 5)
17             else:
18                 a = np.argmax(q[s,:])
19             reward, next_state, done = env(s, a)
20             q[s, a] = q[s, a] + alpha*(reward+gamma*np.max(q[next_state, :])-q[s, a])
21             s = next_state
22             if done:
23                 break
24     return q
```

Para reforzar el entendimiento de este algoritmo se ha proporcionado un **vídeo** donde se explica gráficamente a partir de un juego el funcionamiento de este tipo de aprendizaje por refuerzo.

2.1.3. Aplicaciones del Proceso de Decisión de Markov y Q-Learning en distintas áreas de la robótica móvil.

Tras esta explicación teórica se ha llevado a cabo una investigación exhaustiva sobre distintas aplicaciones del MDP y Q-Learning en la robótica móvil.

En el siguiente **paper** se aplica el algoritmo MDP junto con Q-Learning para tener un control sobre la navegación de un dron. En él se puede obsrvar como se ha conseguido un 88,7 % de éxito en las pruebas realizadas, frente a un 11,3 % de pruebas donde el dron ha colisionado con algún obstáculo.

En el **TFG** que se presenta, se emplea el algoritmo Q-Learning para coordinar un enjambre de drones. Se explica como tras varias iteraciones del algoritmo, los drones son capaces de reconocer una serie de patrones que maximizan el beneficio del resultado.

También se ha encontrado otro **TFG** donde en este caso se ha aplicado el MDP para diseñar un modelo de control de un robot hexápodo. Esto ha permitido que el robot se pueda desplazar manteniéndose estable bajo distintos ambientes.

2.2. Conclusión

El aprendizaje por refuerzo es un enfoque muy utilizado en la robótica, esto se debe en gran parte a que se puede aplicar en básicamente cualquier situación, para cualquier tipo de robot, se puede introducir en un manipulador para que realice tareas de pick and place sin una programación especifica, sin embargo, aunque funcione no es la manera mas efectiva de realizar estas tareas. Por otro lado, en la robótica móvil el aprendizaje por refuerzo si que proporciona una forma efectiva de afrontar los diferentes problemas que se encontrará un coche autónomo, un dron o cualquier vehículo, y esto se debe a la flexibilidad que tiene a la hora de tomar decisiones en escenarios donde nunca se dará la misma situación dos veces.

3. Algoritmos Genéticos

Los algoritmos genéticos, al igual que el aprendizaje por refuerzo, se basan en la forma en la que los seres vivos viven y evolucionan. Como su nombre indica, los algoritmos genéticos son una técnica de programación que imita la evolución de los seres vivos como estrategia para resolver problemas de optimización.

Esta técnica utiliza la selección natural para modelar su estructura de entrenamiento, de manera que se sigue *"la ley del mas fuerte"* donde los individuos mas prometedores, serán los que pasen su *"conocimiento"* a las siguientes generaciones.

3.1. Como funcionan los Algoritmos Genéticos

Los algoritmos genéticos funcionan de manera que hacen que una población de individuos, o conjunto de soluciones, evolucionen sometiéndose a acciones completamente aleatorias, para seguidamente ser evaluados. Esta evaluación es la que determinara si esta nueva *mutación* es prometedora y pasa a las siguientes generaciones.

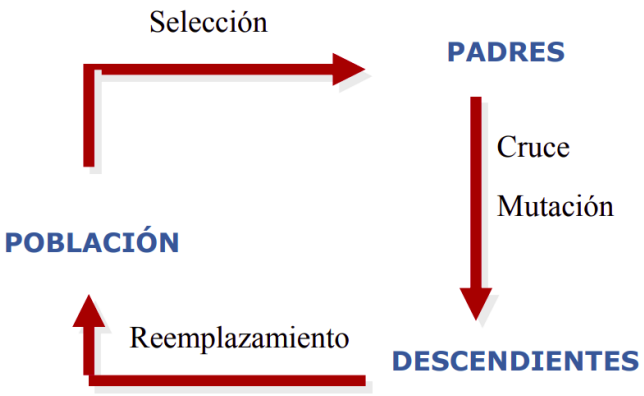


Figura 3: Esquema algoritmo genético.

Con este modelo se consigue un grupo de soluciones, las cuales han sido seleccionadas generación por generación por ser las mas prometedores. Esta selección se realiza en base a un objetivo que se quiera conseguir, el cual marca que acciones serán recompensadas en mayor medida y cuales no.

Estos algoritmos son predominantemente utilizados en problemas de optimización debido a que es necesario generar cientos o miles de distintas soluciones que exploren caminos distintos, de manera que necesitamos que el entorno en el que se trabaja permite comprobar un gran numero de casos.

En este caso también se proporciona un **vídeo** donde se explica los Algoritmos Genéticos con un ejemplo gráfico.

3.2. Algoritmos Genéticos en la robótica móvil

Debido a que estos algoritmos necesitan crear innumerables instancias del problema en cada generación, no es posible aplicar estos algoritmos en un entorno real, ya que comprobar cada individuo de cada generación, uno tras otro, en un mismo robot no es viable.

Sin embargo, si recurrimos a la simulación de nuestro entorno de trabajo, es posible crear una copia de este, en la que podremos generar todas las pruebas de todas las generaciones, de forma mucho menos costosa temporal mente.

Cuando aplicamos los algoritmos genéticos a la robótica móvil, es posible realizar todo el proceso por medio de estos algoritmos, desde que el robot aprenda como debe moverse, simplemente dándole el control de los motores y que el algoritmo aprenda a utilizarlos, hasta que una vez aprendido el movimiento, buscar el mejor camino hasta un objetivo.

A continuación se mostraran varias aplicaciones de estos algoritmos en la robótica móvil:

1. **Araña robótica con aprendizaje de la marcha**

   En este vídeo se muestra como se ha utilizado un algoritmo genético, para que un robot cuadrúpedo aprende a moverse, partiendo solo del control de los motores que controlan sus piernas. Mediante el uso de un acelerómetro se obtiene la posición del robot, y con el objetivo de que esta posición varíe, el robot realiza una serie de movimientos aleatorios hasta que consigue realizar una marcha controlada.

2. **Algoritmo Genético aplicado a un juego estilo *flappy bird***

   Este ejemplo, aunque algo mas distante de la robótica móvil, muestra como el algoritmo va avanzando por las generaciones y van prevaleciendo los individuos que obtienen un resultado mejor.

3. **Algoritmos genéticos en un juego de conducción**

   Por último, en este vídeo se aplica un algoritmo genético para conseguir que un coche simulado recorra un circuito sin que se choque, y en un tiempo mínimo. En este vídeo se aprecia como en la primeras generaciones, los distintos individuos no consiguen avanzar mas que escasos metros mientras que las ultimas generaciones, los coches recorren el circuito en un tiempo aceptable.

4. Enlace al vídeo de presentación.

`https://youtu.be/3hPkW1ve1x4`

Referencias

1. https://ccc.inaoep.mx/~emorales/Papers/2018/2018ESucar.pdf
2. https://oa.upm.es/71417/1/TFM_JUAN_PABLO_MIGUEZ_MELON.pdf
3. https://repositorio.uniandes.edu.co/bitstream/handle/1992/48862/u833488.pdf?sequence=1&isAllowed=y
4. https://www.aprendemachinelearning.com/aprendizaje-por-refuerzo/
5. https://www.ugr.es/~bioestad/_private/cpfund10.pdf
6. https://www.codificandobits.com/curso/aprendizaje-por-refuerzo-nivel-basico/18-ecuaciones
7. https://blog.damavis.com/aprendizaje-por-refuerzo-q-learning/
8. https://blog.adrianistan.eu/reinforcement-learning-aprendizaje-refuerzo-q-learning-3
9. https://repositorio.uniandes.edu.co/bitstream/handle/1992/53431/24390.pdf?sequence=1
10. https://riunet.upv.es/bitstream/handle/10251/134490/Masanet%20-%20Desarrollo%20de%20un%20simulador%20para%20la%20coordinaci%C3%B3n%20de%20drones%20usando%20la%20plataforma%20SPADE.pdf?sequence=1&isAllowed=y
11. https://inaoe.repositorioinstitucional.mx/jspui/bitstream/1009/588/1/CuayaSG.pdf
12. https://www.youtube.com/watch?v=qhRNvCVVJaA&ab_channel=deeplizard
13. https://www.youtube.com/watch?v=RBrXGyo0kIw&ab_channel=BitBoss